

PROGRAM : 1

Warmup-1 > sleepIn

The parameter weekday is true if it is a weekday, and the parameter vacation is true if we are on vacation. We sleep in if it is not a weekday or we're on vacation. Return true if we sleep in.

```
sleepIn(false, false) → true
sleepIn(true, false) → false
sleepIn(false, true) → true
```

Go Save, Compile, Run (ctrl+enter) Show Solution

```
public boolean sleepIn(boolean weekday, boolean Vacation) {
    if (!weekday || Vacation) {
        return true;
    }
    else
        return false;
}
```

Go

Expected	Run
sleepIn(false, false) → true	true OK
sleepIn(true, false) → false	false OK
sleepIn(false, true) → true	true OK
sleepIn(true, true) → true	true OK

✓ All Correct

next | chance

Java > Warmup-1

done page

Code is saved so long as this session is active. Create an account above to save code past this session.

Your progress graph for this problem

PROGRAM :2

Warmup-1 > monkeyTrouble

We have two monkeys, a and b, and the parameters aSmile and bSmile indicate if each is smiling. We are in trouble if they are both smiling or if neither of them is smiling. Return true if we are in trouble.

```
monkeyTrouble(true, true) → true
monkeyTrouble(false, false) → true
monkeyTrouble(true, false) → false
```

Go Save, Compile, Run (ctrl+enter) Show Solution

```
public boolean monkeyTrouble(boolean aSmile, boolean bSmile) {
    if(aSmile && bSmile){
        return true ;
    }
    if(!aSmile && !bSmile){
        return true ;
    }
    else
        return false ;
}
```

Go

Expected	Run
monkeyTrouble(true, true) → true	true OK
monkeyTrouble(false, false) → true	true OK
monkeyTrouble(true, false) → false	false OK
monkeyTrouble(false, true) → false	false OK

✓ All Correct

next | chance

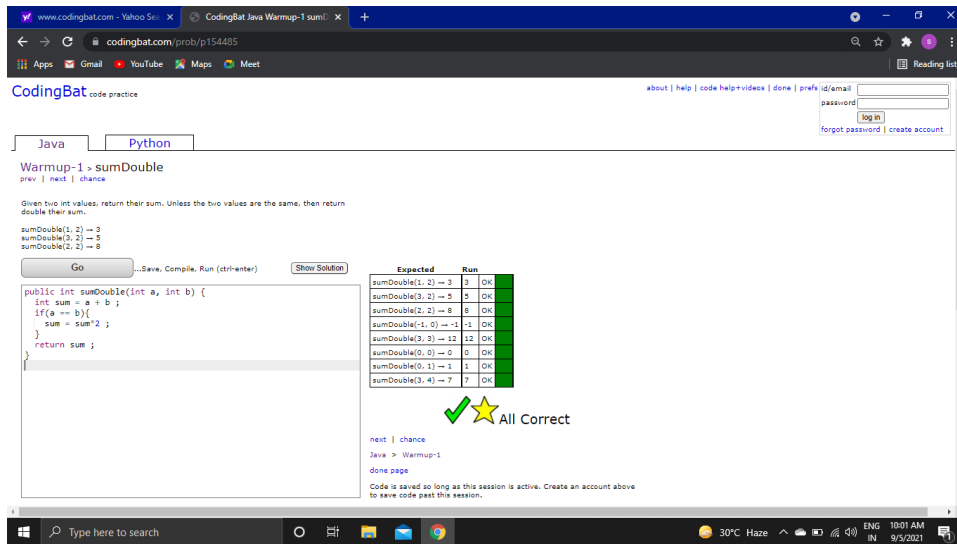
Java > Warmup-1

done page

Code is saved so long as this session is active. Create an account above to save code past this session.

Your progress graph for this problem

PROGRAM : 3



The screenshot shows the CodingBat website interface for the Java Warmup-1 problem 'sumDouble'. The problem description states: 'Given two int values, return their sum. Unless the two values are the same, then return double their sum.' Examples provided are: sumDouble(1, 2) → 3, sumDouble(2, 2) → 5, and sumDouble(2, 2) → 8. The user has written a Java solution in the editor, and the test results on the right show all 10 test cases passing with a green checkmark and 'All Correct' message.

Java Python

Warmup-1 > sumDouble
prev | next | chance

Given two int values, return their sum. Unless the two values are the same, then return double their sum.

sumDouble(1, 2) → 3
sumDouble(2, 2) → 5
sumDouble(2, 2) → 8

Go Save, Compile, Run (ctrl+enter) Show Solution

```
public int sumDouble(int a, int b) {  
    int sum = a + b ;  
    if(a == b){  
        sum = sum*2 ;  
    }  
    return sum ;  
}
```

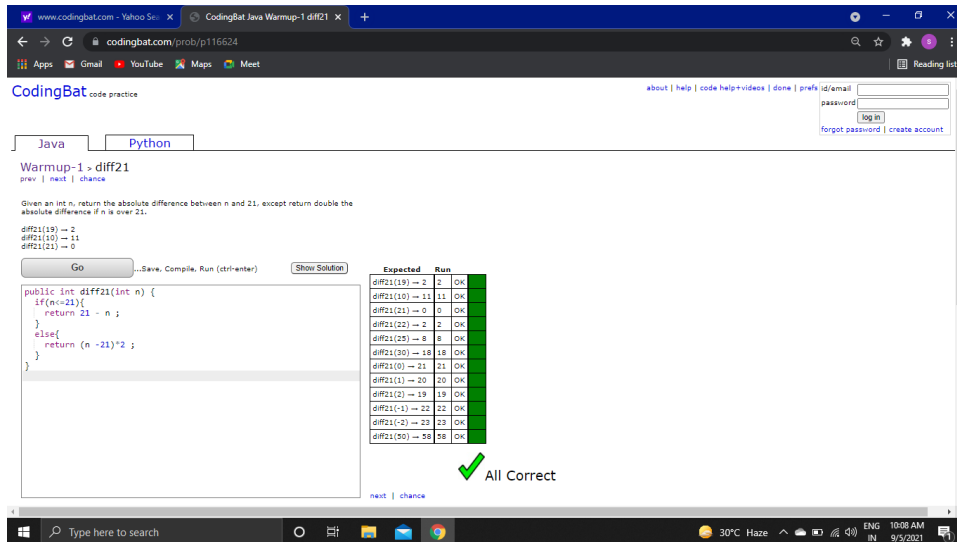
Expected	Run
sumDouble(1, 2) → 3	OK
sumDouble(2, 2) → 5	OK
sumDouble(2, 2) → 8	OK
sumDouble(-1, 0) → -1	OK
sumDouble(3, 2) → 12	OK
sumDouble(0, 0) → 0	OK
sumDouble(0, 1) → 1	OK
sumDouble(3, 4) → 7	OK

✓ All Correct

next | chance
Java > Warmup-1
done page

Code is saved so long as this session is active. Create an account above to save code past this session.

PROGRAM :4



The screenshot shows the CodingBat website interface for the Java Warmup-1 problem 'diff21'. The problem description states: 'Given an int n, return the absolute difference between n and 21, except return double the absolute difference if n is over 21.' Examples provided are: diff21(19) → 2, diff21(10) → 11, and diff21(21) → 0. The user has written a Java solution in the editor, and the test results on the right show all 16 test cases passing with a green checkmark and 'All Correct' message.

Java Python

Warmup-1 > diff21
prev | next | chance

Given an int n, return the absolute difference between n and 21, except return double the absolute difference if n is over 21.

diff21(19) → 2
diff21(10) → 11
diff21(21) → 0

Go Save, Compile, Run (ctrl+enter) Show Solution

```
public int diff21(int n) {  
    if(n<21){  
        return 21 - n ;  
    }  
    else{  
        return (n -21)*2 ;  
    }  
}
```

Expected	Run
diff21(19) → 2	OK
diff21(10) → 11	OK
diff21(21) → 0	OK
diff21(22) → 2	OK
diff21(25) → 6	OK
diff21(30) → 18	OK
diff21(0) → 21	OK
diff21(1) → 20	OK
diff21(2) → 19	OK
diff21(-1) → 22	OK
diff21(-2) → 23	OK
diff21(50) → 58	OK

✓ All Correct

next | chance

PROGRAM :5

The screenshot shows the CodingBat website interface for the 'parrotTrouble' problem. The problem description states: 'We have a loud talking parrot. The "hour" parameter is the current hour time in the range 0..23. We are in trouble if the parrot is talking and the hour is before 7 or after 20. Return true if we are in trouble.' The provided Java solution is:

```
public boolean parrotTrouble(boolean talking, int hour) {
    return (talking && (hour < 7 || hour > 20));
}
```

 The test cases table shows 12 cases, all of which passed with 'OK' status. A green checkmark and the text 'All Correct' are displayed below the table. The interface also includes a 'Go' button, a 'Show Solution' button, and a 'done page' link.

Expected	Run
parrotTrouble(true, 6) → true	true OK
parrotTrouble(true, 7) → false	false OK
parrotTrouble(false, 6) → false	false OK
parrotTrouble(true, 21) → true	true OK
parrotTrouble(false, 21) → false	false OK
parrotTrouble(false, 20) → false	false OK
parrotTrouble(true, 23) → true	true OK
parrotTrouble(false, 23) → false	false OK
parrotTrouble(true, 20) → false	false OK
parrotTrouble(true, 12) → false	false OK

PROGRAM :6

The screenshot shows the CodingBat website interface for the 'makes10' problem. The problem description states: 'Given 2 ints, a and b, return true if one if them is 10 or if their sum is 10.' The provided Java solution is:

```
public boolean makes10(int a, int b) {
    return (a == 10 || b == 10 || a + b == 10);
}
```

 The test cases table shows 12 cases, all of which passed with 'OK' status. A green checkmark and the text 'All Correct' are displayed below the table. The interface also includes a 'Go' button, a 'Show Solution' button, and a 'done page' link.

Expected	Run
makes10(9, 10) → true	true OK
makes10(9, 9) → false	false OK
makes10(1, 9) → true	true OK
makes10(10, 1) → true	true OK
makes10(10, 10) → true	true OK
makes10(8, 2) → true	true OK
makes10(8, 3) → false	false OK
makes10(10, 42) → true	true OK
makes10(12, -2) → true	true OK

PROGRAM :7

Given an int n, return true if it is within 10 of 100 or 200. Note: Math.abs(num) computes the absolute value of a number.

nearHundred(93) → true
nearHundred(90) → true
nearHundred(89) → false

Go Save, Compile, Run (ctrl-enter) Show Solution

```
public boolean nearHundred(int n) {  
    return ((Math.abs(100 - n) <= 10) ||  
           (Math.abs(200 - n) <= 10));  
}
```

Go

Editor font size % (125) Shorter output

Expected	Run
nearHundred(93) → true	true OK
nearHundred(90) → true	true OK
nearHundred(89) → false	false OK
nearHundred(110) → true	true OK
nearHundred(111) → false	false OK
nearHundred(121) → false	false OK
nearHundred(101) → false	false OK
nearHundred(209) → false	false OK
nearHundred(190) → true	true OK
nearHundred(200) → true	true OK
nearHundred(0) → false	false OK
nearHundred(5) → false	false OK
nearHundred(50) → false	false OK
nearHundred(191) → true	true OK
nearHundred(189) → false	false OK
nearHundred(200) → true	true OK
nearHundred(110) → true	true OK
nearHundred(211) → false	false OK
nearHundred(290) → false	false OK

✓ All Correct

next | chance

Java > Warmup-1

PROGRAM : 8

Given 2 int values, return true if one is negative and one is positive. Except if the parameter "negative" is true, then return true only if both are negative.

posNeg(1, -1, false) → true
posNeg(1, 1, false) → true
posNeg(-4, -5, true) → true

Go Save, Compile, Run (ctrl-enter) Show Solution

```
public boolean posNeg(int a, int b, boolean negative) {  
    if(negative){  
        return (a < 0 && b < 0);  
    }  
    else {  
        return ((a < 0 && b > 0) || (a > 0 && b < 0));  
    }  
}
```

Go

Editor font size % (125) Shorter output

Expected	Run
posNeg(1, -1, false) → true	true OK
posNeg(-1, 1, false) → true	true OK
posNeg(-4, -5, true) → true	true OK
posNeg(-4, -5, false) → false	false OK
posNeg(-4, 5, false) → true	true OK
posNeg(-4, 5, true) → false	false OK
posNeg(1, 1, false) → false	false OK
posNeg(-1, -1, false) → false	false OK
posNeg(1, 1, true) → false	false OK
posNeg(-1, 1, true) → false	false OK
posNeg(1, 1, true) → false	false OK
posNeg(-1, -1, true) → true	true OK
posNeg(5, -5, false) → true	true OK
posNeg(6, 6, false) → true	true OK
posNeg(-5, -6, false) → false	false OK
posNeg(-2, -2, false) → false	false OK
posNeg(1, 2, false) → false	false OK
posNeg(-5, 6, true) → false	false OK
posNeg(-5, -5, true) → true	true OK

✓ All Correct

next | chance

Java > Warmup-1

PROGRAM :9

The screenshot shows the CodingBat website interface for the 'notString' problem. The problem description states: 'Given a string, return a new string where "not" has been added to the front. However, if the string already begins with "not", return the string unchanged. Note: use equals() to compare 2 strings.' The user has provided a Java solution, and the test cases have been successfully executed, resulting in 'All Correct'.

Expected vs Run Test Results:

Expected	Run
notString("candy") == "not candy"	OK
notString("not") == "not not"	OK
notString("not bad") == "not bad"	OK
notString("bad") == "not bad"	OK
notString("not") == "not"	OK
notString("is not") == "not is not"	OK
notString("no") == "not no"	OK

```
public String notString(String str) {
    if (str.length() >= 3 && str.substring(0, 3).equals("not")) {
        return str;
    }
    return "not " + str;
}
```

PROGRAM :10

The screenshot shows the CodingBat website interface for the 'missingChar' problem. The problem description states: 'Given a non-empty string and an int n, return a new string where the char at index n has been removed. The value of n will be a valid index of a char in the original string (i.e. n will be in the range 0..str.length()-1 inclusive).' The user has provided a Java solution, and the test cases have been successfully executed, resulting in 'All Correct'.

Expected vs Run Test Results:

Expected	Run
missingChar("litten", 1) == "litten"	OK
missingChar("litten", 0) == "litten"	OK
missingChar("litten", 4) == "litten"	OK
missingChar("Hi", 0) == "i"	OK
missingChar("Hi", 1) == "H"	OK
missingChar("code", 0) == "ode"	OK
missingChar("code", 1) == "ode"	OK
missingChar("code", 2) == "ode"	OK
missingChar("code", 3) == "cod"	OK
missingChar("chocolate", 8) == "chocolat"	OK

```
public String missingChar(String str, int n) {
    String front = str.substring(0, n);
    String back = str.substring(n+1, str.length());
    return front + back;
}
```